User's Guide to the HPC-Systems at ZIH

Version 2.3

Ulf Markwardt

November 14, 2007

Disclaimer. This booklet is mainly directed at the users of the new HPC-systems. However users of our smaller Opteron- and Itanium-clusters may find it usefull, too.
SGI, Altix, and Origin are registered trademarks of Silicon Graphics International. Other brands and names may be claimed a the property of others.

This manuscript is work in progress, since we try to incorporate more information with increasing experience and with every question you ask us. Please tell us if you miss something or find incorrect information. You can find this document and further information at the web site
http://www.tu-dresden.de/zih → Publikationen → Schriften → Benutzerinformationen.

Acknowledgements. I would like to thank the following people for contributing to this manual: Matthias Müller, Reiner Vogelsang (SGI), Matthias Jurenz, Matthias Lieber, Guido Juckeland, Michael Kluge, Robert Henschel

Contents

1 Introduction **5**

2 Hardware **6**
2.1 HPC Component SGI Altix . . . 6
2.1.1 ccNuma Architecture . . . 6
2.1.2 Compute Module . . . 6
2.1.3 CPU . . . 7
2.2 Linux Networx PC-Farm Deimos . . . 8
2.2.1 CPU . . . 8
2.3 Linux Networx PC-Cluster Phobos . . . 8
2.3.1 CPU . . . 9

3 Operating Systems **9**
3.1 Login . . . 9
3.2 Customize your environment . . . 9
3.3 Backup . . . 10
3.4 Batch Systems . . . 10
3.4.1 Interactive Jobs . . . 11
3.4.2 Monitoring . . . 11
3.4.3 Parallel Jobs . . . 12
3.4.4 Placing Threads or Processes on CPUs . . . 13

4 Software Development **14**
4.1 Compilers . . . 15
4.1.1 Compiler Flags . . . 15
4.2 Parallel Programming . . . 17
4.2.1 MPI . . . 17
4.2.2 OpenMP . . . 17
4.3 Debuggers . . . 17
4.3.1 Allinea DDT . . . 18
4.3.2 Intel Debugger (idb) . . . 18
4.3.3 GNU Debugger (gdb/ddd) . . . 18
4.4 Performance Tuning . . . 19
4.4.1 Basics . . . 19
4.4.2 Analyzing Profiles . . . 19
4.4.3 Determining Data Access Patterns . . . 20
4.4.4 Vampir . . . 20
4.5 Mathematical Libraries . . . 22
4.5.1 Math Kernel Library (MKL) . . . 23
4.5.2 ACML . . . 23
4.5.3 ATLAS . . . 24
4.5.4 SGI SCSL . . . 24
4.5.5 FFTW . . . 24
4.6 Miscellaneous . . . 24
4.6.1 I/O with from/to binary files . . . 24
4.6.2 Fast I/O on Altix . . . 24
4.6.3 Memory Corruption on Altix . . . 25

5 Applications **26**
5.1 Quantum Chemistry, Molecular Modeling . . . 26
5.1.1 Gaussian . . . 26
5.1.2 CPMD . . . 26
5.1.3 NAMD . . . 27
5.2 Bioinformatics . . . 27
5.2.1 PHYLIP . . . 27
5.2.2 CLUSTALW . . . 27
5.2.3 HMMER . . . 27
5.2.4 NCBI ToolKit . . . 28
5.3 Engineering . . . 28
5.3.1 Abaqus . . . 28
5.3.2 Ansys . . . 28
5.3.3 Ansys CFX . . . 29
5.3.4 Fluent . . . 29
5.4 Mathematics . . . 29
5.4.1 MATLAB . . . 29
5.4.2 Mathematica . . . 30
5.4.3 Maple . . . 30

6 Support from ZIH **31**
6.1 Support Requests . . . 31

7 Further Documentation **32**
7.1 SGI developer forum . . . 32
7.2 OpenMP . . . 32
7.3 MPI . . . 32
7.4 Intel Itanium . . . 32
7.5 Libraries and Compilers . . . 33
7.6 Tools . . . 33

A Appendix **34**
A.1 Problems with Intel Compilers . . . 34

1 Introduction

The Center for Information Services and High Performance Computing (ZIH) is a central scientific unit of TU Dresden with a strong competence in parallel computing and software tools. We have a strong commitment to support *real users*, collaborating to create new algorithms, applications and to tackle the problems that need to be solved to create new scientifc insight with computational methods. Our new compute complex "Hochleistungs-Rechner-/-Speicher-Komplex" (HRSK) is focused on data-intensive computing. High scalability, big memory and fast I/O-systems are the outstanding properties of this project, aside from the significant performance increase (cf. fig 1).
The infrastructure is provided not only to TU Dresden but to all universities and research institutes in Saxony.

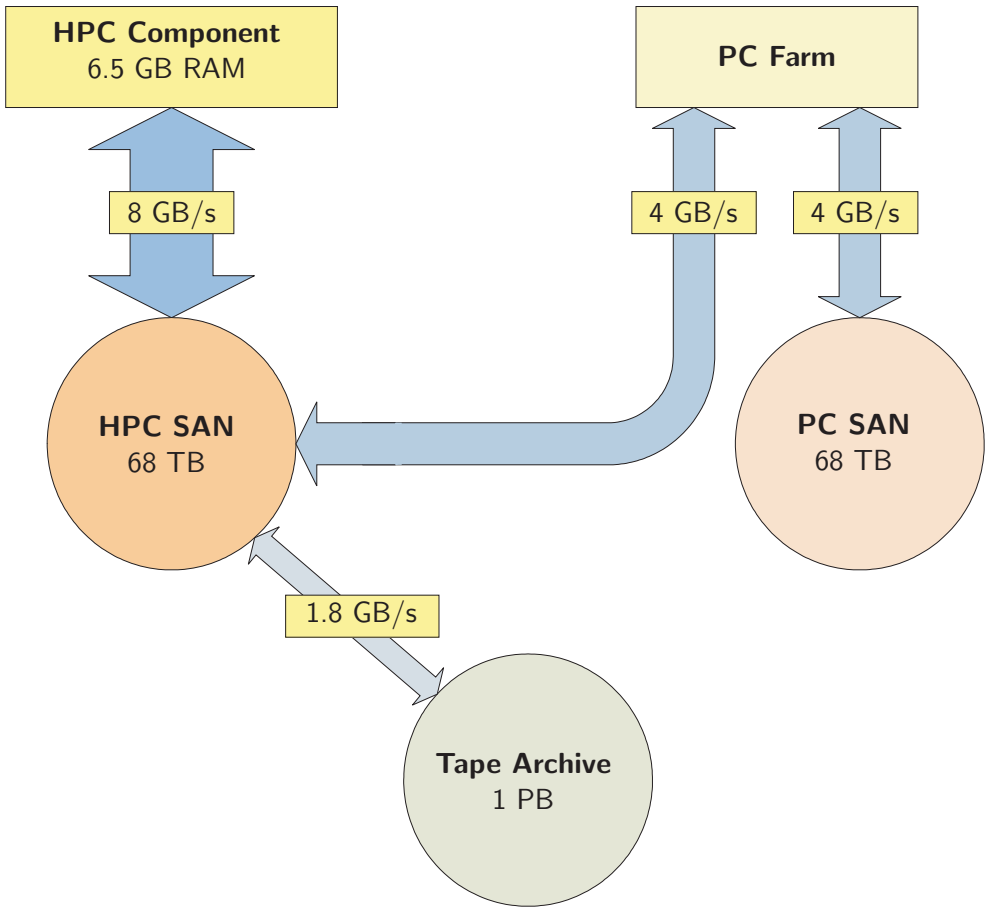

Figure 1: Overview over the HRSK-system

2 Hardware

This chapter should provide you with basic information about the hardware installed at ZIH between 2005 and 2007.

2.1 HPC Component SGI Altix

The SGI Altix 4700 is a shared memory system with dual core Intel Itanium 2 CPUs (Montecito) operated by the Linux operating system SuSE SLES 10 with a 2.6 kernel. Currently, the following Altix partitions are installed at ZIH:

name	total cores	compute cores	Memory per core
Mars	384	348	1 GB
Jupiter	512	506	4 GB
Saturn	512	506	4 GB
Uranus	512	506	4 GB
Neptun	128	128	1 GB

The jobs for these partitions (except *Neptun*) are scheduled by a LSF batch system running on `mars.hrsk.tu-dresden.de`. The actual placement of a submitted job may depend on factors like memory size, number of processors, time limit (cf. chapter 3.4). All partitions share the same CXFS filesystem.

2.1.1 ccNuma Architecture

The SGI Altix has a ccNUMA architecture, which stands for Cache Coherent Non-Uniform Memory Access. It can be considered as a SM-MIMD (*shared memory - multiple instruction multiple data*) machine. The SGI ccNuma system has the following properties:

- Memory is physically distributed but logically shared
- Memory is kept coherent automatically by hardware.
- Coherent memory: memory is always valid (caches hold copies)
- Granularity is L3 cacheline (128 B)
- Bandwidth of NumaLink4 is 6.4 GB/s

The ccNuma is a compromise between a distributed memory system and a flat symmetric multi processing machine (SMP). Altough the memory is shared, the access properties are not the same.

2.1.2 Compute Module

The basic compute module of an Altix system is shown in fig. 2. It consists of one dual core Intel Itanium 2 "Montecito" processor, the local memory of 4 GB (2 GB on *Mars*), and the communication component, the so-called SHUB. All resources are shared by both cores. They have a common front side bus, so that accumulated memory bandwidth for both is not higher than for just one core.
The SHUB connects local and remote ressources. Via the SHUB and NUMAlink all CPUs can access remote memory in the whole system. Naturally, the fastest access provides local memory (fig. 3). There are some hints and commands that may help you

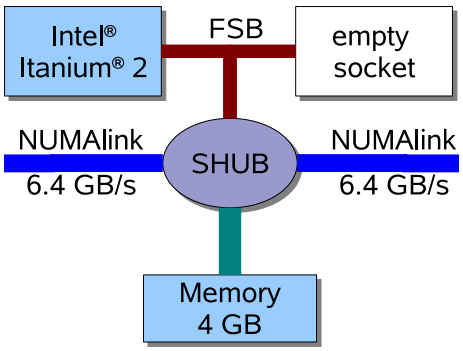

Figure 2: Altix compute blade

to get optimal memory allocation and process placement (cf. chapter 3.4.4.1). Four of these blades are grouped together with a NUMA router in a compute brick. All bricks are connected with NUMAlink4 in a "fat-tree"-topology.

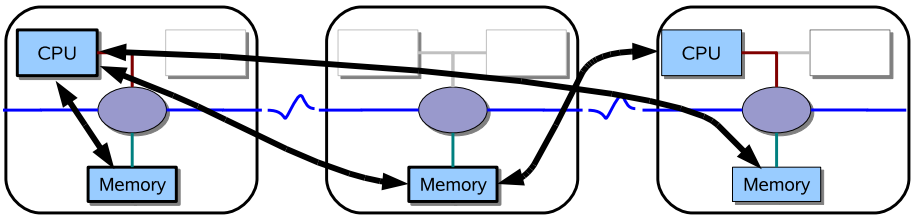

Figure 3: Remote memory access via SHUBs and NUMAlink

2.1.3 CPU

The current SGI Altix is based on the dual core Intel Itanium 2 processor (codename "Montecito"). One core has the following basic properties:

clock rate	1.6 GHz
integer units	6
floating point units (multiply-add)	2
→ peak performance	6.4 GFLOPS
L1 cache	2 x 16 kB, 1 clock latency
L2 cache	256 kB, 5 clock latency
L3 cache	9 MB, 12 clock latency
front side bus	128 bit x 200 MHz

The theoretical peak performance of all Altix partitions is hence about 13.1 TFLOPS.

The processor has hardware support for efficient software pipelining. For many scientific applications it provides a high sustained performance exceeding the performance of RISC CPUs with similar peak performance. On the down side is the fact that the compiler has to explicetly discover and exploit the parallelism in the application.

2.2 Linux Networx PC-Farm Deimos

The PC farm *Deimos* is a heterogenous cluster based on dual core AMD Opteron CPUs. The nodes are operated by the Linux operating system SuSE SLES 10 with a 2.6 kernel. Currently, the following hardware is installed:

CPUs	AMD Opteron X85 dual core
RAM per core	2 GB
Number of cores	2584
total peak performance	13.4 TFLOPS
single chip nodes	384
dual nodes	230
quad nodes	88
quad nodes (32 GB RAM)	24

All nodes share a 68 TB filesystem on DDN hardware. Each node has per core 40 GB local diskspace for scratch mounted on `/tmp`. The jobs for the compute nodes are scheduled by a LSF batch system from the login nodes `deimos.hrsk.tu-dresden.de`. Two separate Infiniband networks (10 Gb/s) with low cascading switches provide the communication and I/O infrastructure for low latency / high throughput data traffic. An additional gigabit Ethernet network is used for control and service purposes.
Users with a login on the Altix can access their home directory via NFS below the mount point `/hpc_work`.

2.2.1 CPU

The cluster is based on dual-core AMD Opteron X85 processor. One core has the following basic properties:

clock rate	2.6 GHz
floating point units	2
→ peak performance	5.2 GFLOPS
L1 cache	2 x 64 kB
L2 cache	1 MB
memory bus	128 bit x 200 MHz

The CPU belongs to the x86_64 family. Since it is fully capable of running x86-code, one should compare the performances of the 32 and 64 bit versions of the same code.

2.3 Linux Networx PC-Cluster Phobos

Phobos is a cluster based on AMD Opteron CPUs. The nodes are operated by the Linux operating system SuSE SLES 9 with a 2.6 kernel. Currently, the following hardware is installed:

CPUs	AMD Opteron 248 (single core)
total peak performance	563.2 Gflops
# nodes	64 compute + 1 master
CPUs per node	2
RAM per node	4 GB

All nodes share a 4.4 TB SAN filesystem. Each node has additional local diskspace mounted on `/scratch`. The jobs for the compute nodes are scheduled by a LSF batch system running on the login node `phobos.hrsk.tu-dresden.de`.
Two seperate Infiniband networks (10 Gb/s) with low cascading switches provide the infrastructure for low latency / high throughput data traffic. An additional GB/Ethernetwork is used for control and service purposes.

2.3.1 CPU

Phobos is based on single-core AMD Opteron 248 processor. It has the following basic properties:

clock rate	2.2 GHz
floating point units	2
→ peak performance	4.4 GFLOPS
L1 cache	2 x 64 kB
L2 cache	1 MB
memory bus	128 bit x 200 MHz

The CPU belongs to the x86_64 family. Although it is fully capable of running x86-code, one should always try to use 64-bit programs due to their potentially higher performance.

3 Operating Systems

Make sure you know how to work with a Linux system. Documentations and tutorials can be easily found in the internet or in your library.

3.1 Login

The only way to login to the machines is via `ssh`[1]. From a Linux console, the command syntax is `ssh [<user>@]<host>`. The option `-X` enables X11 forwarding for graphical applications. The default shell is `bash`.

Hostname	Description
`mars.hrsk.tu-dresden.de`	SGI Altix 4700 - LSF
`neptun.hrsk.tu-dresden.de`	SGI Altix 4700 - with FPGA and graphic hardware
`deimos.hrsk.tu-dresden.de`	Linux Networx PC Farm
`phobos.hrsk.tu-dresden.de`	Linux Networx PC Cluster

3.2 Customize your environment

To allow the user to switch between different versions of installed programs and libraries we use the so called *module* concept. A module is a user interface that provides utilities for the dynamic modification of a user's environment, i.e., users do not have to manually modify their environment variables (`PATH`, `LD_LIBRARY_PATH`, ...) to access the compilers, loader, libraries, and utilities.

[1] For security reasons, this port is only accessible for hosts within the domains of TU Dresden. Guests from other research institutes can either use one of the central login servers or the VPN gateway of ZIH. Information on these topics can be found on our web pages http://www.tu-dresden.de/zih.

Command	Description
`module help`	show all module options
`module list`	list all user-installed modules
`module porge`	remove all user-installed modules
`module avail`	list all available modules
`module load <modname>`	load module `modname`
`module switch <mod1> <mod2>`	unload module `mod1`; load module `mod2`

Please note we have set `ulimit -c 0` as a default to prevent you from filling the disk with the dump of a crashed program. `bash`-users can use `ulimit -c unlimited` to enable the debugging via analyzing the core file (`limit coredumpsize unlimited` for `tcsh`).

3.3 Backup

An automated backup system provides security for the HOME-directories on *Mars*, *Deimos*, and *Phobos* on a daily basis.

This is the reason why we urge our users to store (large) temporary data (like checkpoint files) on the `/fastfs`-Filesystem or at local scratch disks.

3.4 Batch Systems

Both HRSK systems are operated with the batch system LSF running on *Mars* and *Deimos*, resp. The job submission can be done with the command: `bsub [bsub_options] <job>`

Some options of `bsub` are shown in the following table:

bsub option	Description
`-n <N>`	set number of processors to N(default=1)
`-W <hh:mm>`	set maximum wallclock time to ¡hh:mm¿
`-R "rusage[mem=MEM_MB]"`	needed memory size in MB
`-J <name>`	assigns the specified name to the job
`-eo <errfile>`	writes the standard error output of the job to the specified file (**o**verwriting)
`-o <outfile>`	appends the standard output of the job to the specified file
`-R "span[hosts=1]"`	use only one SMP node (for OpenMP jobs!)
`-x`	disable other jobs to share the node (*Deimos*).

You can use the `%J`-macro to merge the job ID into names.

It is more convenient to put the options directly in a job file you can submit with

```
bsub  < my_jobfile
```

An example job file may look like this:

```
#!/bin/bash
#BSUB -W 4:00                          # max. wall clock time 4h
#BSUB -R "rusage[mem=1500]"            # memory for the job in MB
#BSUB -R "span[hosts=1]"               # run on a single node
#BSUB -n 8                             # number of processors
#BSUB -o out.%J                        # output file
#BSUB -u vorname.nachname@tu-dresden.de    # email address

  echo Starting Program
  cd $HOME/work
  a.out                                # e.g. an OpenMP program
  echo Finished Program
```

LSF sets the user environment according to the environment at the time of submission. Based on the given information the job scheduler puts your job into the appropriate queue. These queues are subject to permanent changes. You can check the current situation using the command `bqueues -l`.
There are a couple of rules and restrictions to balance the system loads. One idea behind them is to prevent users from occupying the machines unfairly. An indicator for the priority of a job placement in a queue is therefore the ratio between used and granted CPU time for a certain period.

3.4.1 Interactive Jobs

Interactive activities like editing, compiling etc. are normally limited to the boot CPU set (*Mars*) or to the master nodes (*Deimos*). If you want to start a parallel interactive job you again have to use the batch system.
Use the additional bsub options `-Is` to start an interactive job like:

```
bsub -Is matlab
```

You can check the current usage of the system with the command `bhosts` to estimate the time to schedule.

3.4.2 Monitoring

The command `bhosts` shows the load on the hosts. For a little more information, use `lsf_info` (in `/usr/local/bin`) for a short summary on the job situation on the system. For a more convenient overview the command `showjobs` displays information on the LSF status like this:

```
You have 1 running job using 64 cores
You have 1 pending job
-----------------------------------+-------------------------------------------
nodes available:         714/714         nodes damaged:                 0
-----------------------------------+-------------------------------------------
jobs running:            1797        |    cores closed (exclusive jobs): 94
jobs wait:               3361        |    cores closed by ADMIN:         129
jobs suspend:            0           |    cores working:                 2068
jobs damaged:            0           |
-----------------------------------+-------------------------------------------
normal working cores:    2556            cores free for jobs:           265
```

With the command `bqueues [-l <queuename>]` you can get information about available queues. With `bqueues -l` you get a detailed listing of the queue properties.
The command `bjobs` allows to monitor your running jobs. It has the following options:

bjobs option	Description
-r	Displays running jobs.
-s	Displays suspended jobs, together with the suspending reason that caused each job to become suspended.
-p	Displays pending jobs, together with the pending reasons that caused each job not to be dispatched during the last dispatch turn.
-a	Displays information on jobs in all states, including finished jobs that finished recently.
-l [job_id]	Displays detailed information for each job or for a particular job.

3.4.3 Parallel Jobs

For submitting parallel jobs, a few rules have to be understood and followed. In general they depend on the type of parallelization and the architecture.

3.4.3.1 OpenMP Jobs An SMP-parallel job can only run within a node (or a partition), so it is neccessary to include the option `-R "span[hosts=1]"`. The maximum number of processors for an SMP-parallel program is 506 on an Altix partition, and 8 on a quad node on *Deimos*. A simple example of a job file for an OpenMP job can be found above (section 3.4).

3.4.3.2 MPI Jobs There are major differences for submitting MPI-parallel jobs on the systems. Please refer to chapter 4.2.1 for compiling MPI programs. It is essential to use the same modules at compile- and run-time.

Mars
The MPI library running on the Altix is provided by SGI and highly optimized for the ccNUMA architecture of this machine. However, communication within a partition is faster than across partitions. Take this into consideration when you submit your job.

- Single-partition jobs can be started like this:

```
bsub -R "span[hosts=1]" -n 16 mpirun -np 16 a.out
```

- Really large jobs with over 256 CPUs might run over multiple partitions. Cross-partition jobs can be submitted via PAM like this

```
bsub -n 1024 pamrun a.out
```

Deimos
Most MPI implementations on "normal" clusters communicate via ethernet fabrics. On *Deimos* (and *Phobos*), we have a high-bandwidth, low-latency Infiniband network for communication. Yet, it is a bit tricky to handle from the user's point of view. Per default, when you specify a compiler module, the corresponding OpenMPI library can

be used without much trouble. E.g. `module load pathscale` changes the environment so, that you can use `mpicc` to compile and link MPI parallel C codes built with `pathcc`. (`mpiCC` and `mpif90` for C++ and Fortran90 codes, resp.)
Please pay attention to the messages you get loading the module. They are more up-to-date than this manual.
To submit a job the user has to use a script or a command-line like this:

```
bsub -n <N> -a openmpi mpirun.lsf a.out
```

Phobos
Per default, when you specify a compiler module, the corresponding MVAPICH library is loaded automatically. E.g. `module load pgi` changes the environment so, that you can use `mpicc` to compile and link MPI parallel C codes built with `pathcc`. (`mpiCC` and `mpif90` for C++ and Fortran90 codes, resp.)
Please pay attention to the messages you get loading the module. They are more up-to-date than this manual.
To submit a program (compiled with PGI compiler) the user has to use a script or a command-line like this:

```
module load pgi # if not already loaded
bsub -n <N> -a mvapich mpirun.lsf a.out
```

You can switch the MPI library manually like

```
module load pgi # if not already loaded
module switch mvapich_pgi openmpi_pgi
bsub -n <N> -a openmpi mpirun.lsf a.out
```

3.4.4 Placing Threads or Processes on CPUs

3.4.4.1 dplace on Altix To bind threads to CPUs you can use the `dplace` command. Important flags are:

Flag	Description
`-c <cpulist>`	CPU numbers are logical numbers relative to current cpumemset.
`-x <mask>`	A bitmask for specifying threads to skip placing. [See following examples.]
`-s <count>`	Skip placement of the first ¡count¿ threads. Use s1 to skip placing the shepherd thread in MPI programs.
`-q`	Displays static load information. dplace without arguments will avoid loaded cpus.
`-e`	Exact placement

When you run OpenMP applications you have to be aware that the run time library uses the second thread for internal management purposes. You therefore need to use:

```
dplace -x 2 -c 0-<N>
```

The use of profiling tools may require modification of placement flags.

```
dplace -x5 -c0-15 histx -o prof a.out
        histx                          skip          (1)
        a.out master                   place         (0)
        OpenMP monitor                 skip          (1)
        a.out slave1                   place         (0)
        a.out slave2                   place         (0)
        ...                            place         (0)
```

You can use dplace in conjunction with MPI:

```
mpirun -np <#> dplace -s 1 ./a.out
```

An easier approach is to set the environment variable

```
export MPI_DSM_DISTRIBUTE=1
```

3.4.4.2 taskset on Farm and Cluster To place tasks on the PC farm, you can use the standard linux tool `taskset` which allows to "bond" a process to a given set of CPUs on the system. The Linux scheduler will honor the given CPU affinity and the process will not run on any other CPUs.
This only makes sense when you submit the job with `bsub -x ...` to exclusively use the hosts.
For forther information on `taskset` please refer to the man pages.

4 Software Development

This section should provide you with the basic knowledge and tools to get you out of trouble. It will tell you:

- How to compile your code
- Using mathematical libraries
- Find caveats and hidden errors in application codes
- Handle debuggers
- Follow system calls and interrupts
- Understand the relationship between correct code and performance

Some hints that are helpful:

- Stick to standards wherever possible. Computers are short living creatures, migrating between platforms can be painful. In addition, running your code on different platforms greatly increases the reliabily. You will find many bugs on one platforms that never will be revealed on another.
- Before and during performance tuning: Make sure that your code delivers the correct results.

Some questions you should ask yourself:

- Given that a code is parallel, are the results independent from the numbers of threads or processes?
- Have you ever run your Fortran code with array bound and subroutine argument checking `-check all -traceback`?
- Have you checked that your code is not causing floating point exceptions?
- Does your code work with a different link order of objects?
- Have you made any assumptions regarding storage of data objects in memory?

4.1 Compilers

The following compilers are available[2] on our platforms:

	Intel 10 icc icpc ifort	**GNU 4.1** gcc g++ gfortran	**PGI 7.0** pgcc pgCC pgf95	**Pathscale 3.0** pathcc pathCC pathf95
Mars	x	((x))		
Deimos	x	x	x	x
Phobos	9.1	4.1 6.2	2.4	

GNU compilers are installed on the Altix but they reach significantly less performance on IA64. Please do not use them without urgency.

All C compiler support ANSI C and C99 with a couple of different language options. The support for Fortran77, Fortran90, Fortran95, and Fortran2003 differs from one compiler to the other. Please check the man pages to verify that your code can be compiled.

Please note that the linking of C++ files normally requires the C++ version of the compiler to link the correct libraries. For serious problems with Intel's compilers please refer to Appendix A.1, p. 34.

4.1.1 Compiler Flags

Common options are:

- `-g` to include information required for debugging
- `-pg` to generate `gprof`-style sample-based profiling information during the run
- `-O<0|1|2|3>` to customize the optimization level from no (`-O0`) to aggressive (`-O3`) optimization
- `-I` to set search path for header files
- `-L` to set search path for libraries

Please note that aggressive optimization allows deviation from the strict IEEE arithmetic. Since the performance impact of options like `-mp` is very hard the user herself has to balance speed and desired accuracy of her application. There are several options for profiling, profile-guided optimization, data alignment and so on. You can list all available compiler options with the option `--help`. Reading the man-pages is a good idea, too.

The user benefits from the (nearly) same set of compiler flags for optimization for the C,C++, and Fortran-compilers. In the following table, only a couple of important compiler-dependant options are listed. For more detailed information, the user should refer to the man pages or use the option `--help` to list all options of the compiler.

[2] Use `module avail` to list the installed versions on the platform. The names of these modules may change without further notice.

Intel	PGI	Pathscale	Description
-openmp	-mp	-mp	turn on OpenMP support
-mp	-Kieee	-no-fast-math	use this flag to limit floating-point optimizations and maintain declared precision
-mp1	-Knoieee	-ffast-math	some floating-point optimizations are allowed, less performance impact than -mp.
-fpe<n>[3] -ftz[4]	-Ktrap=...		Controls the behavior of the processor when floating-point exceptions occur.
	-tp amd64	-mcpu=opteron	optimize for Opteron processor
-axW	-fastsse	-msse2	"generally optimal flags" for supporting SSE instructions (Opteron only)
-ipo	-Mipa	-ipa	inter procedure optimization (across files)
-ip	-Mipa		inter procedure optimization (within files)
-parallel	-Mconcur	-apo	Auto-parallelizer
-prof-gen	-Mpfi	-fb-create <FN>	Create intrumented code to generate profile in file <FN>
-prof-use	-Mpfo	-fb-opt <FN>	Use profile data for optimization. - Leave all other optimization options

We can not generally give advice as to which option should be used - even -O0 sometimes leads to a fast code. To gain maximum performance please test the compilers and a few combinations of optimization flags. In case of doubt, you can also contact ZIH and ask the staff for help.

[3] ifort only

[4] flushes denormalized numbers to zero: On Itanium 2 an underflow raises an underflow exception that needs to be handled in software. This takes about 1000 cycles!

4.2 Parallel Programming

4.2.1 MPI

4.2.1.1 Mars This installation of the Message Passing Interface supports the MPI 1.2 standard with a few MPI-2 features (see `man mpi`). There is no command like mpicc, instead you just have to append `-lmpi` to the linker command line. Since the include files as well as the library are in standard directories there is no need to append additional library- or include-paths. Note for C++ programmers: You additionally need to link with `-lmpi++`.

4.2.1.2 Deimos When loading a compiler module on *Deimos*, the module for the MPI implementation OpenMPI is also loaded. Use the wrapper commands `mpicc`, `mpiCC`, `mpif77`, and `mpif90` to compile MPI source code. They use the currently loaded compiler. To reveal the commandlines behind the wrappers, use the option `--show`. For running your code, you have to load the same compiler module as for compiling the program.

4.2.2 OpenMP

To achieve the best performance the compiler needs to exploit the parallelism in the code. Therefore it is sometimes necessary to provide the compiler with some hints. Some possible directives are (Fortran style):

`CDEC$ ivdep`	ignore assumed vector dependences
`CDEC$ swp`	try to software-pipeline
`CDEC$ noswp`	disable softeware-pipeling
`CDEC$ loop count (NN)`	hint for optimzation
`CDEC$ distribute point`	split this large loop
`CDEC$ unroll (n)`	unroll (n) times
`CDEC$ nounroll`	do not unroll
`CDEC$ prefetch a`	prefetch array a
`CDEC$ noprefetch c`	do not prefetch array a

The compiler directives are the same for `ifort` and `icc`. The syntax for C/C++ is like `#pragma ivdep`, `#pragma swp`, and so on. More detailled sources of information are listed in chapter 7.4.

4.3 Debuggers

This short User's Guide only describes how to start the debuggers on our HPC systems. For detailed information refer to 7.6.
General advices for debugging are:

- You need to compile your code with the flag `-g` to enable debugging. It is also recommendable to reduce or even disable optimizations (`-O0`).
- For parallel applications try to reconstruct the problem with less processes before using a debugger. DDT behaves slower with larger number of processes.
- The flag `-traceback` of the Intel Fortran compiler causes to print stack trace and source code location when the program terminates abnormally.

4.3.1 Allinea DDT

The Allinea Debugger is availble with `module load ddt`. It is quite intuitive usable and provides great support for MPI-parallel applications. For serial applications run DDT with:

```
ddt [PROGRAM [PROGRAM ARGS]]
```

Select "'none"' as MPI implementation in the DDT session control window (via the "Change" button) and start the program with the "Run" button.

4.3.1.1 Debugging MPI applications on `Mars`

For parallel applications on *Mars*, replace `mpirun -np <N>` in your job submission by `ddt`, e.g.:

```
bsub -W 1:00 -I -n 8 ddt ./a.out
```

Select "altix-mpi" as MPI implementation and set the number of processes. Click "Run" to start the program. Alternatively, you can start an interactive batch session (a shell on compute CPUs) and start DDT from there. This is especially useful when you plan to do several consecutive debug sessions:

```
bsub -W 1:00 -Is -n 8 bash
ddt ./a.out
```

4.3.1.2 Debugging MPI applications on `Deimos`

Using DDT on *Deimos* is similar to *Mars*. For OpenMPI, submit the job the following:

```
bsub -x -W 1:00 -I -n 8 -a openmpi ddt ./a.out
```

Then set the number of processes in the DDT session control window and click the "Submit" button. DDT also works in an interactive batch session on the compute nodes:

```
bsub -W 1:00 -Is -n 8 -a openmpi bash
ddt ./a.out
```

4.3.2 Intel Debugger (idb)

The Intel debugger (available with `module load idb`) can be used for programs compiled with an Intel compiler.

```
idb [-dbx] [-gdb] [-pid process_id] [exec_file [core_file]]
```

4.3.3 GNU Debugger (gdb/ddd)

This debugger offers only limited support for MPI parallel application and Fortran90. However, it might be the debugger you are most used to:

```
gdb [exec_file [core_file|process_id]]
```

The graphical frontend `ddd` for `gdb` can be used after `module load ddd`.

```
ddd [-debugger name] [exec_file [core_file|process_id]
```

4.4 Performance Tuning

4.4.1 Basics

There are many possible reasons for performance problems. This chapter is only intended to provide a short overview and some entry points where to start.

- CPU-bound processes
  - Performing many "slow" operations sqrt, fp divides
  - Non-pipelined operations
  - Switching between adds and mults
- Memory-bound processes
  - Poor memory strides
  - Page thrashing
  - Cache misses
  - Poor data placement (in NUMA systems)
- I/O bound processes
  - Performing synchronous I/O
  - Performing formatted I/O
  - Library and system level buffering

4.4.1.1 Floating Point Performance on Itanium 2 The Itanium 2 CPU (Altix) is capable of delivering 2 floating point multiply-adds per clock cycle (i.e. peak performance of 6 GFlops). This performance can slow down due to two reasons: getting data to the processor cannot be done quickly enough or the FP unit is throwing exceptions.
Denormal numbers (underflow numbers) are defined in the IEEE standard for floating point as those below the normal range. Operations involving these numbers cannot be performed on the processor and need to be performed by the operating system and there is a huge penalty (about 1.000 cycles) in doing this.
The system logs some of these events. The user can identify them by using the command `dmesg | grep assist`. She will get events like:

```
namd2(19416): floating-point assist fault at ip 4000000000590941, isr 0000020000000008
namd2(19416): floating-point assist fault at ip 4000000000590941, isr 0000020000000008
namd2(19416): floating-point assist fault at ip 4000000000590941, isr 0000020000000008
namd2(19462): floating-point assist fault at ip 4000000000410941, isr 0000020000000008
```

The commandline tool `addr2line` might help to localize te corresponding source code. Please be aware that its result might be disturbed by optimizations.
If you are sure, the underflows are no problems for your application, you should compile with the option `-ftz`

4.4.2 Analyzing Profiles

A very convenient way to select the focus for further optimization is to analyze the frequencies and durations of function calls. To do this, one has to compile the code with the appropriate flags: `-pg` for Pathscale, PGI or GNU compilers, and `-p` for `icc`
At the end of the execution of the program, the collected data is written into a file. The user can then use a profiling tool like `gprof` or `kprof` to display the data.

4.4.3 Determining Data Access Patterns on Altix

The command `dlook` allows you to display the memory map and CPU usage for a specified process

```
dlook [-a] [-c] [-h] [-l] [-o outfile] [-s secs] command [ command-args]
dlook [-a] [-c] [-h] [-l] [-o outfile] [-s secs] pid
```

For each page in the virtual address space of the process, dlook prints the following information:

- The object that owns the page, such as a file, SysV shared memory, a device driver, etc.
- The type of page, such as random access memory (RAM), FETCHOP, IOSPACE, etc.
- For RAM pages, the following are also listed:
  - memory attributes (SHARED, DIRTY, etc.)
  - node that the page is located on
  - physical address of page, if option `-a` is specified

4.4.4 Vampir

Vampir is a graphical analysis framework that provides a large set of different chart representations of event based performance data generated through source code instrumentation. These graphical displays, including state diagrams, statistics, and timelines, can be used by developers to obtain a better understanding of their parallel programs inner working and to subsequently optimize it. Vampir allows for quick focusing on appropriate levels of detail which allows the detection and explanation of various performance bottlenecks such as load imbalances and communication deficiencies.
The Vampir tool has been developed at the Center for Applied Mathematics of Research Center Jülich and the Center for High Performance Computing of the Technische Universität Dresden. Vampir is available as a commercial product since 1996 and has been enhanced in the scope of many research and development projects. In the past, it was distributed by the German Pallas GmbH which became later a part of Intel Corporation. The cooperation with Intel ended in 2005 but the development is continued by ZIH.

A growing number of performance monitoring environments like VampirTrace (see below), TAU or KOJAK can produce tracefiles that are readable by Vampir. Since version 5.0, Vampir supports the new Open Trace Format (OTF) that is developed by ZIH aswell and is especially designed for massively parallel programs.

A detailed documentation on Vampir can be found at http://www.vampir.eu.

Before using Vampir set up the correct environment with `module load vampir`. Start the GUI with

```
bsub -I vampir
```

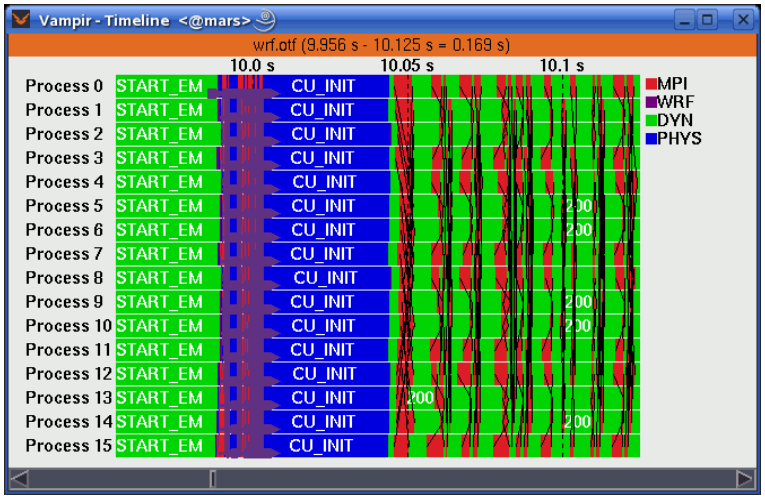

Figure 4: Vampir Global Timeline

4.4.4.1 Vampir-Server on Mars Vampir-Server comes in two parts: a daemon `vngd` is analyzing the tracefiles, and a front-end `vng` provides a GUI. The correct environment can be set with `module load vng`. The daemon is a multithreaded program, it has to be started in a queue like:

```
bsub -n 4 -I vngd -n 4
```

After scheduling this job the daemon prints the number of the port it is serving, like `Listen port: 30088`. If the daemon started in a non-interactive queue (without bsub-option `-I`), then the used port can be determined by looking in the file `$HOME/.vngd`:

```
cat \$HOME/.vngd
```

In another shell the user can (after loading the module `module load vng`) start the front-end with

```
bsub -I vng -a localhost -p 30088
```

Please make sure you shut down the daemon after finishing your work with the front-end!

4.4.4.2 VampirTrace VampirTrace is a performance monitoring tool, that produces tracefiles during a program run. These tracefiles can be analyzed and visualized by the tool Vampir (see above). Before using VampirTrace, set up the correct environment with `module load vampirtrace`.
To make measurements with VampirTrace, the user's application program needs to be instrumented, i.e., at specific important points ("events") VampirTrace measurement calls have to be activated. By default, VampirTrace handles this automatically. In order to enable instrumentation of function calls, MPI as well as OpenMP events, the user only needs to replace the compiler and linker commands with VampirTrace's wrappers. The following list shows some examples depending on the parallelization type of the program:

- **Serial programs**: Compiling serial code is the default behavior of the wrappers. Simply replace the compiler by VampirTrace's wrapper:

  original: `gfortran a.f90 b.f90 -o myprog`
  with instrumentation: `vtf90 a.f90 b.f90 -o myprog`

  This will instrument user functions (if supported by compiler) and link the VampirTrace library.

- **MPI parallel programs**: If your MPI implementation uses MPI compilers (this is the case on *Deimos* and *Phobos*), you need to tell VampirTrace's wrapper to use this compiler instead of the serial one:

  original: `mpicc hello.c -o hello`
  instrumented: `vtcc -vt:cc mpicc hello.c -o hello`

  MPI implementations without own compilers (as on the Altix) require the user to link the MPI library manually. In this case, you simply replace the compiler by VampirTrace's compiler wrapper:

  original: `icc hello.c -o hello -lmpi`
  instrumented: `vtcc hello.c -o hello -lmpi`

  If you want to instrument MPI events only (creates smaller trace files and less overhead) use the option `-vt:inst manual` to disable automatic instrumentation of user functions.

- **OpenMP parallel programs**: When VampirTrace detects OpenMP flags on the command line, OPARI is invoked for automatic source code instrumentation of OpenMP events:

  original: `ifort -openmp pi.f -o pi`
  instrumented: `vtf77 -openmp pi.f -o pi`

- **Hybrid MPI/OpenMP parallel programs**: With a combination of the above mentioned approaches, hybrid applications can be instrumented:

  original: `mpif90 -openmp hybrid.F90 -o hybrid`
  instrumented: `vtf90 -vt:f90 mpif90 -openmp hybrid.F90 -o hybrid`

By default, running a VampirTrace instrumented application should result in a tracefile in the current working directory where the application was executed.
Consult the documentation for more detailed information (e.g. manual source code instrumentation, important environment variables, recording hardware counter by using PAPI library, memory allocation tracing, I/O tracing, function filtering and grouping).
The installed documentation can be found on *Mars* and *Deimos* in the folder `/licsoft/tools/vampirtrace/<VERSION>/share/vampirtrace/doc`.

4.5 Mathematical Libraries

The following mathematical are available on our platforms (including the two seperate clusters):

	Mars	*Deimos*	*Phobos*
MKL	8.1	8.1	9.1
ACML		3.6	3.6
ATLAS		3.6	3.6
SCSL	1.6.1		

4.5.1 Math Kernel Library (MKL)

The Intel Math Kernel Library is a collection of basic linear algebra subroutines (BLAS) and fast fourier transformations (FFT). It contains routines for:

- Solvers such as linear algebra package (LAPACK) and BLAS
- Eigenvector/eigenvalue solvers (BLAS, LAPACK)
- PDEs, signal processing, seismic, solid-state physics (FFTs)
- General scientific, financial - vector transcendental functions, vector markup language (VML)

More specifically it contains the following components:

- BLAS:
  - Level 1 BLAS: vector-vector operations, 48 functions
  - Level 2 BLAS: matrix-vector operations, 66 functions
  - Level 3 BLAS: matrix-matrix operations, 30 functions
- LAPACK (linear algebra package), solvers and eigensolvers, hundreds of routines, more than 1000 user callable routines
- FFTs (fast Fourier transform): one and two dimensional, with and without frequency ordering (bit reversal). There are wrapper functions to provide an interface to use MKL instead of FFTW.
- VML (vector math library), set of vectorized transcendental functions
- Parallel Sparse Direct Linear Solver (Pardiso)

Please note: MKL comes in a OpenMP-parallel version. If you want to use it, make sure you know how to place your jobs (chapter 3.4.4.1).

4.5.2 ACML

Opteron based systems only.

The AMD Core Math Library is a collection of the following routines:

- A full implementation of Level 1, 2 and 3 Basic Linear Algebra Subroutines (BLAS), with key routines optimized for high performance on AMD Opteron processors.
- A full suite of Linear Algebra (LAPACK) routines. As well as taking advantage of the highly-tuned BLAS kernels, a key set of LAPACK routines has been further optimized to achieve considerably higher performance than standard LAPACK implementations.
- A comprehensive suite of Fast Fourier Transforms (FFTs) in both single-, double-, single-complex and double-complex data types.
- Fast scalar, vector, and array math transcendental library routines optimized for high performance on AMD Opteron processors.
- Random Number Generators in both single- and double-precision.

4.5.3 ATLAS

The ATLAS (Automatically Tuned Linear Algebra Software) project is an ongoing research effort focusing on applying empirical techniques in order to provide portable performance. At present, it provides C and Fortran77 interfaces to a portably efficient BLAS implementation, as well as a few routines from LAPACK.

4.5.4 SGI SCSL

For the SGI Altix, SCSL provides similar functionality as the Intel MKL. One advantage is that there is a version for programs requiring 64 Bit integers. The SCSL routines can be linked and loaded by using the `-lscs` or the `-lscs_mp` options. Try `man scsl` for more information.

4.5.5 FFTW

FFTW is a C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discrete cosine/sine transforms or DCT/DST).
Before using this library, please check out the functions of vendor specific libraries ACML and/or MKL.

4.6 Miscellaneous

4.6.1 I/O with from/to binary files

This section is only important for users migrating from other architectures.
The Itanium and Opteron CPUs use so called little-endian order to store numbers (most significant byte with highest memory address). So if you access files written in binary mode on big endian platforms they may not work on an Itanium platform without conversion.
Due to the little endian representation you normally have to convert binary files written on big endian architectures before reading them. For Fortran applications there is however the option to do this conversion automatically.
Big endian systems are SGI MIPS/Irix (Origin 3000, 2000,...), HP PA Risc, Sun Sparc, IBM Power RISC, NEC vector systems, Cray vector systems.
For the use with Intel-compilers you can read/write big-endian binary data by the following means:

- set the environment variable `F_UFMTENDIAN=big` (applies to all units) or `F_UFMTENDIAN=big:10,20` (applies to unit 10 and 20 only) or
- compile with `-convert big` (Intel compilers) or `-Mbyteswapio` (PGI, Pathscale)

4.6.2 Fast I/O on Altix

The ffread and ffwrite functions provide flexible file I/O (FFIO) to record-oriented or byte stream-oriented data in an application-transparent manner (see `man ffread`).

4.6.3 Memory Corruption on Altix

The `MALLOC_CHECK_` environent variable controls some basic protection against memory corruption (see `man malloc`):

Value	Description
0	silently ignore any heap corruption
1	print diagnostic message when heap corruption detected
2	abort immediately upon heap corruption

This only detects simple errors such as one-byte overruns and multiple free() calls.

5 Applications

The following applications are available on the HRSK systems. (General descriptions are taken from the vendor's web site or from Wikipedia.org.)
Before running an application you normally have to load the given module (e.g. `module load ansys`). Please read the instructions given while loading th emodule, they are more up-to-date than this manual.

5.1 Quantum Chemistry, Molecular Modeling

5.1.1 Gaussian

Version: G03
Vendor: http://www.gaussian.com
Module: gaussian
Machines: *Deimos*

General Starting from the basic laws of quantum mechanics, Gaussian predicts the energies, molecular structures, and vibrational frequencies of molecular systems, along with numerous molecular properties derived from these basic computation types. It can be used to study molecules and reactions under a wide range of conditions, including both stable species and compounds which are difficult or impossible to observe experimentally such as short-lived intermediates and transition structures.
To be able to run Gaussian, you have to be in the user group gauss. To check this, use the Linux command `groups`, which lists all groups you are a member of.
With "module load gaussian" you can set the environment according to the needs of Gaussian. For temporary data (GAUSS_SCRDIR), please use your own /fastfs/... directory!
We have a queue named gauss, which can be used for time intensive computing that can not be checkpointed.
One can submit a Gaussian job with options like this:

```
bsub -n 1 \ ,
-R "span[hosts=1] rusage[mem=MEM_MB]" \
-x \
-m NODE_TYPE \

-W hh:mm \
script.sh
```

`MEM_MB`... memory usage in MB
(for exclusive usage)
`NODE_TYPE` can be `single_hosts`, `dual_hosts`, `quad_hosts`, or `fat_quads`
needed wallclock time

5.1.2 CPMD

Version: 3.11
Vendor: http://www.cpmd.org
Module: cpmd
Machines: *Deimos*

General The Car-Parrinello Molecular Dynamics, better known as CPMD, is a package for performing ab-initio quantum mechanical molecular dynamics (MD) using pseudopotentials and a plane wave basis set. This code is a parallelized implementation of Density functional theory.

Please submit a parallel job like this

```
bsub -n 32 -a openmpi -o out mpirun.lsf cpmd.x bp_110_wf.inp
```

or a sequential job with:

```
bsub cpmd.seq bp_110_wf.inp
```

5.1.3 NAMD

Version: 2.6
Vendor: http://www.ks.uiuc.edu/Research/namd
Module: namd
Machines: *Deimos*

General NAMD is a parallel molecular dynamics code designed for high-performance simulation of large biomolecular systems.

On *Deimos*, NAMD scales well up to 64 CPUs depending on the size of the problem. You can use a call like

```
bsub -a openmpi -x -oo output -n 32 mpirun.lsf namd\_2.6\_tcl run100.conf
```

5.2 Bioinformatics

5.2.1 PHYLIP

Version: 3.66
Vendor: J. Felsenstein; University of Washington
Module: phylip
Machines: *Deimos*, *Mars*

General PHYLIP (the PHYLogeny Inference Package) is a package of programs for inferring phylogenies (evolutionary trees). Methods that are available in the package include parsimony, distance matrix, and likelihood methods, including bootstrapping and consensus trees. Data types that can be handled include molecular sequences, gene frequencies, restriction sites and fragments, distance matrices, and discrete characters. CLUSTALW is automatically loaded together with the PHYLIP module in order to create PHYLIP input data.

5.2.2 CLUSTALW

Version: 1.83
Vendor: http://www.ebi.ac.uk/clustalw
Module: clustalw
Machines: *Deimos*, *Mars*

General Multiple alignments of protein sequences are important tools in studying sequences. The basic information they provide is the identification of conserved sequence regions. This is very useful in designing experiments to test and modify the function of specific proteins, in predicting the function and structure of proteins and in identifying new members of protein families. ClustalW is a general purpose multiple sequence alignment program for DNA or proteins. It produces biologically meaningful multiple sequence alignments of divergent sequences. It calculates the best match for the selected sequences, and lines them up so that the identities, similarities and differences can be seen.

5.2.3 HMMER

Version: 2.3.2
Vendor: http://hmmer.janelia.org
Module: hmmer / hmmer-pthread
Machines: *Deimos*, *Mars*

General Profile hidden Markov models (profile HMMs) can be used to do sensitive database searching using statistical descriptions of a sequence family's consensus. HMMER is a freely distributable implementation of profile HMM software for protein sequence analysis.
The PThread version of HMMER should be used with 2 CPUs. Using more than two CPUs will not improve performance. Make sure that the number of CPUs that is specified in the `bsub` call is identical to the number of CPUs that is specified in the command line parameter when calling

```
hmmpfam
```

5.2.4 NCBI ToolKit

Version: 3.66
Vendor: http://www.ncbi.nlm.nih.gov
Module: ncbitoolkit
Machines: *Deimos*, *Mars*

General Molecular biology is generating a host of data which are dramatically altering and deeping our understanding of the processes which underlie all living things. This new knowledge is already affecting medicine, agriculture, biotechnology, and basic science in fundamental and sweeping ways. However, the data on which our growing understanding is based is being accumulated and analyzed in thousands of laboratories all over the world, from large genome centers to small university laboratories, from large pharmacutical companies to small biotech startups. It is being managed and analyzed on machines from small personal computers to supercomputers, on systems from a few disk files to large commercial database systems. These essential new data require specialized tools for analysis and management, so software tools are being developed in all these different environments at once.The GenInfo Software Toolbox is a set of software and data exchange specifications that are used by NCBI to produce portable, modular software for molecular biology.

5.3 Engineering

5.3.1 Abaqus

Version: ABAQUS 6.6.1
Vendor: http://www.hks.com
Module: abaqus
Machines: *Deimos*

General ABAQUS is a general-purpose finite-element program designed for advanced linear and nonlinear engineering analysis applications with facilities for linking in user developed material models, elements, friction laws.

5.3.2 Ansys

Version: Ansys 11.0
Vendor: http://www.ansys.com
Module: ansys
Machines: *Deimos*

General ANSYS is a general-purpose finite-element program for engineering analysis, and includes preprocessing, solution, and postprocessing functions. ANSYS is used in a wide range of disciplines for solutions to mechanical, thermal, and electronic problems.

Please do not run Ansys on the login node. Use a call like this for a pure computation:

```
bsub -n <N> -R "span[hosts=1]" ansys110 -np <N> -b -p aa_t_a -o <output.txt> -i <input.txt>
```

If your problem needs the "Research" license, substitute aa_t_a by aa_r. The usage of more than (N=)4 CPUs is not advisable (by CADFEM). Make sure to include `-R "span[hosts=1]"` in your job submission. - Ansys only runs on a single SMP node. For interactive jobs, please submit Ansys via the batch system like:

```
bsub ansys110 -g -p aa_t_a
```

5.3.3 Ansys CFX

Version: 10.0
Vendor: http://www.ansys.com
Module: cfx
Machines: *Deimos*

General ANSYS CFX is a powerful finite-volume-based program package for modelling general fluid flow in complex geometries. The main components of the CFX package are the flow solver cfx5solve, the geometry and mesh generator cfx5pre, and the post-processor cfx5post.

5.3.4 Fluent

Version: 6.3.26
Vendor: http://www.fluent.com
Module: fluent
Machines: *Deimos*

General Fluent is a general purpose package for modeling fluid flow and heat transfer. It can simulate two-/three-dimensional, steady/unsteady, compressible/incompressible flows in structured and unstructured grids. The Mach number of Fluent simulation ranges from subsonic to hypersonic. Its capabilities include simulating non-isothermal flows, disperse phase/droplets, combustion and radiation heat transfer, and flow-through porous media.
For parallel jobs, we have provided a wrapper function fluent.lsf which adds the necesary options for parallel runs over multiple nodes. You can start a compute session like this

```
bsub -oo out.txt -oe error.txt -x -n 4 -m dual\_hosts fluent.lsf 3d -g -i INPUTFILE
```

5.4 Mathematics

5.4.1 MATLAB

Version: 2007b
Vendor: http://www.mathworks.com
Module: matlab
Machines: *Deimos*

General MATLAB is a numerical computing environment and programming language. Created by The MathWorks, MATLAB allows easy matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs in other languages. Although it specializes in numerical computing, an optional toolbox interfaces with the Maple symbolic engine, allowing it to be part of a full computer algebra system.
To use Matlab in an interactive session, please submit your job with `bsub -Is matlab`.

5.4.2 Mathematica

Version: 6.0
Vendor: http://www.wolfram.com
Module: mathematica
Machines: *Deimos*

General Mathematica is a general computing environment, organizing many algorithmic, visualization, and user interface capabilities within a document-like user interface paradigm.
To use Mathematica in an interactive session, please submit your job with `bsub -Is mathematica`.

5.4.3 Maple

Version: 11
Vendor: http://www.maplesoft.com
Module: maple
Machines: *Deimos*

General Maple is an all purpose mathematics software tool. Maple provides an advanced, high performance mathematical computation engine with fully integrated numerics and symbolics, all accessible from a WYSIWYG technical document environment. Live math is expressed in its natural 2D typeset notation, linked to state-of-the-art graphics and animations with full document editing and presentation control.
To use Maple in an interactive session, please submit your job with `bsub -Is maple` or `bsub -Is xmaple`.

6 Support from ZIH

Over the last 10 years since the founding of the former Center for High Performance Compution (ZHR) the staff has been supporting users, developing tools, and collecting experience in the field of high performance computing. We are currently using the following tools for code instrumentation and analysis:

- Vampir-NG 1.4 - ZIH tool,
- Intel VTune
- UPI - Universal Profiling Interface tool in development by ZIH.

If you think what your application needs is a little speed-up, don't hesitate to ask the authors to organize some support.
Experience tells that during the code development phase you are in constant need for help to make your program run correctly. For a leading edge computational science code it is normal to be under constant development.

6.1 Support Requests

The status of our machines and messages concerning maintainance shutdowns etc. can be found at http://www.tu-dresden.de/zih/aktuelles/betriebsstatus.
For support requests and other questions regarding HPC the email address hpcsupport@zih.tu-dresden.de has been established. This email address is served by a trouble ticket system.

7 Further Documentation

You can find detailed documentation in the `doc` - directory of the installed products, e.g. `/opt/intel/cc_90/doc`.

At the web site http://www.tu-dresden.de/zih → Publikationen → Schriften → Benutzerinformationen you can find these links, further information, and updates.
The most recent information is available at the web sites of our machines at http://tu-dresden.de/zih/hrsk/.

7.1 SGI developer forum

The web sites behind http://www.sgi.com/developers/resources/tech_pubs.html are full of most detailed information on SGI systems. Have a look onto the section 'Linux Publications'. You will be redirected to the public part of SGI's technical publication repository.

- Linux Application Tuning Guide
- Linux Programmer's Guide, The
- Linux Device Driver Programmer's Guide
- Linux Kernel Internals.... and more.

7.2 OpenMP

You will find a lot of information at the following web pages:

- http://www.openmp.org
- http://www.compunity.org

7.3 MPI

The following sites may be interesting:

- http://www.mcs.anl.gov/mpi/ - the MPI homepage.
- http://www.mpi-forum.org/ - Message Passing Interface (MPI) Forum Home Page
- http://www.open-mpi.org/ - the dawn of a new standard for a more fail-tolerant MPI.

The manual for SGI-MPI (installed on *Mars*) can be found at:
http://techpubs.sgi.com/library/manuals/3000/007-3773-003/pdf/007-3773-003.pdf

7.4 Intel Itanium

. There is a lot of additional material regarding the Itanium CPU:

- http://www.intel.com/design/itanium/manuals/iiasdmanual.htm
- http://www.intel.com/design/archives/processors/itanium/index.htm
- http://www.intel.com/design/itanium2/documentation.htm

You will find the following manuals:

- Intel Itanium Processor Floating-Point Software Assistance handler (FPSWA)
- Intel Itanium Architecture Software Developer's Manuals Volume 1: Application Architecture
- Intel Itanium Architecture Software Developer's Manuals Volume 2: System Architecture
- Intel Itanium Architecture Software Developer's Manuals Volume 3: Instruction Set
- Intel Itanium 2 Processor Reference Manual for Software Development and Optimization
- Itanium Architecture Assembly Language Reference Guide

7.5 Libraries and Compilers

- http://www.intel.com/software/products/mkl/index.htm
- http://www.intel.com/software/products/ipp/index.htm
- http://www.ball-project.org/
- http://www.intel.com/software/products/compilers/ - Intel Compiler Suite
- http://www.pgroup.com/doc - PGI Compiler
- http://pathscale.com/ekopath.html - PathScale Compilers

7.6 Tools

- http://www.allinea.com/downloads/userguide.pdf - Allinea DDT Manual
- http://www.intel.com/software/products/compilers/docs/linux/idb_manual_l.html - Intel Debugger
- http://www.gnu.org/software/gdb/documentation/ - GNU Debugger
- http://vampir-ng.de - official homepage of Vampir, an outstanding tool for performance analysis developed at ZIH.
- http://www.fz-juelich.de/zam/kojak/ - homepage of KOJAK at the FZ Jülich. Parts of this project are used by Vampirtrace.
- http://www.intel.com/software/products/threading/index.htm

A Appendix

A.1 Problems with Intel Compilers

If you encounter a bug in one of the intel compilers we ask you to report this issue to hpcsupport@zih.tu-dresden.de . We have a support contract with Intel to get bugs fixed. Please apply the following procedure to report the bug:

1. Create one single source file:

   C/C++ : use the -E option to produce a single file.
   ```
   icc -E myfile.c > bug.c
   ```

   Fortran : use fgather
   ```
   fgather myfile.F90
   ```

2. Run the compiler under the control of cesr compiler command
   `cesr icc -O3 bug.c` or
   `cesr ifort -O3 cesrgathered.F90`
   Please read the man page for a more detailled description. Be aware that this may take a long time.

3. The tool will print out a summary containing the compiler command required to reproduce the error and the outputfile that was generated. Please provide us with the summary and the outputfile.

This procedure has the following advantages:

1. It protects your intellectual property, because you don't have to send in your complete source code

2. it reduces the amount of code you have to send

3. it makes life easier for the engineers and this will reduce the amount of time required to fix this bug.

Please also report whether you have found a workaround, e.g. using a different compiler version or using different compiler flags. We need to know how critical this issue is for you.